What am I trying to do

- Recreate fl-studio zgamevisualizer (but automatic)
  - i.e, making live music visualizations with computer graphics
  - Shaders are programs that map pixels on screen to colors. They usually have modifiable parameters
  - https://youtu.be/KLtXvvahR8w?t=11480
- How are the shaders made?
  - https://www.shadertoy.com/playlist/NltcWN
  - https://thebookofshaders.com/
- Why?
  - Making visualizations while making music makes things complicated
  - Doesn't translate to live situations
    - The z-game output is just a video, not responsive
  - Alternatives for live situations:
    - Only use spectrograms, not customizable
      - Very old method (windows media player)
    - Not free
    - https://www.virtualdj.com/forums/223454/General_Discussion/Best_visualization_shaders_for_VirtualDJ.html

First Goal: Automatic Visualization with spectrograms

Spectrograms are created with continual ***FFT*** application
- https://musiclab.chromeexperiments.com/spectrogram/

I want to implement something like this in rust
https://www.shadertoy.com/view/llB3W1

Second Goal: Enhance feature detection

- I want to do more interesting feature extraction

    - Would like my own code that I can debug and study for better algorithms
        - Better features: Longer term outlook, Detect instruments, etc

Implementation

Audio spectrogram extraction

Listen to an audio stream (microphone) and extract frequency information in near-real time
- Implemented algorithm on pre-recorded sound
- Implemented algorithm on mic input

Visualization

Make visuals that react to sound

- I wanted to use p5.js
    - https://p5js.org/learn/getting-started-in-webgl-shaders.html
- p5.js (and other libraries) didn't work out. I decided to use rust
    - How would data sharing work? OSC isn't good enough
    - using just rust seems more sustainable
    - I want to have a deeper understanding of computer graphics
- Decided to use a well documented and low-level graphics library (WGPU)
    - https://sotrh.github.io/learn-wgpu/
    - trade-off of extensibility vs ease of use